REMARKS

Claims 1, 6, 10, 13, 17, 32 and 35 have been amended. No claims have been added or cancelled. Claims 1-11, 13-26, 28-35 and 37 are pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Finality of the Rejection:

Applicants respectfully request removal of the finality of the current Office Action because the Examiner has included a new ground of rejection not necessitated by amendment. Specifically, the Examiner has rejected claims 10, 11 and 13-16 under U.S.C. § 112, second paragraph. As stated in the M.P.E.P at 706.07(a), a "second or any subsequent actions on the merits shall be final, *except where the examiner introduces a new ground of rejection that is neither necessitated by applicant's amendment of the claims nor based on information submitted in an information disclosure statement*". The current 35 U.S.C. § 112 rejections of claims 10 and 13 are new grounds of rejection that were not necessitated by any amendment of the claims nor was it based on information submitted in an information disclosure statement. Thus, the finality of the current Office Action is improper and removal thereof is respectfully requested.

Section 112, Second Paragraph, Rejection:

The Examiner rejected claims 10, 11 and 13-16 under 35 U.S.C. § 112, second paragraph, as indefinite. Applicants traverse this rejection and assert that the claims were not indefinite. However, to expedite prosecution, claims 10 and 13 have been amended for clarity. Removal of the § 112 rejection is respectfully requested.

Section 103(a) Rejection:

The Examiner rejected claims 1, 6, 9, 17, 20 and 31 under 35 U.S.C. § 103(a) as being unpatentable over Montero, et al. (U.S. Publication 2002/0143958) (hereinafter

"Montero") in view of Goldick (U.S. Publication 2003/0093457). Applicants respectfully traverse this rejection for at least the reasons below.

Regarding claim 1, contrary to the Examiner's assertion, Montero in view of Goldick does not teach or suggest a distributed store configured to provide locked access to the primary state of session data to a process executing within one of a plurality of application servers, where in providing locked access to the primary state to a process executing within one of the plurality of application servers, the distribute store is configured to send a lock token to the process, wherein only processes that have received a lock token can access the primary state. The Examiner admits that Montero does not teach a distributed store configured to send a lock token and relies upon Goldick, citing paragraphs 24-25 and 44-45. Goldick teaches a system for managing allocation of resources and locks to client computer systems. Goldick's system includes locking resources and maintaining subscriptions to lock-related events to effectively allow for asynchronous grants of a lock based on the time of the request to alleviate lock starvation. Goldick further teaches that a server may break an existing lock to prevent lost resources.

However, Goldick, even if combined with Montero, does not mention anything about a distributed store sending lock tokens to processes *executing within an application server*. Instead, Goldick teaches a server that provides requesting *client* computer systems with locked access to resources. Goldick specifically refers to managing access by client computer systems. Goldick's system for managing access to shared resources by client computer systems is very different from a distributed store sending a lock token to a process executing within an application server and from Montero's system for providing periodic storing of session data by multiple application servers. Goldick does not mention anything about sending a lock token to processes executing within an application server. Nor does Goldick have anything to do with providing a plurality of application servers locked access to session data. Additionally, Goldick is not directed toward providing locked access to *session data* at all. Instead, Goldick teaches a system for managing *client* access to shared resources, such as "text documents, application program modules, data objects, properties or attributes for data objects". The shared

resources taught by Goldick are very different from session data configured for access by a plurality of application servers. The Examiner has not cited any prior art, either singly or combination, that teaches or suggests a distributed store configured to provide locked access to the primary state of session data to a process executing within one of a plurality of application servers, where in providing locked access to the primary state to a process executing within one of the plurality of application servers, the distribute store is configured to send a lock token to the process.

Additionally, the Examiner's proposed combination of Montero and Goldick would not result in a system that included a distributed store configured to send a lock token to a process executing within one of a plurality of application servers. Instead, the Examiner's combination of Montero and Goldick would result in Montero's back-end database system that includes application servers maintaining session data for clients as taught by Montero, but also allows clients locked access to a server database, based on providing requesting clients lock tokens, as taught by Goldick. **As shown above, Montero nor Goldick, whether considered singly or in combination, fail to teach anything regarding a distributed store configured to send a lock token *to a process executing within one of a plurality of application servers.*** Thus, the Examiner's combination of Montero and Goldick does not teach or suggest all the limitations of Applicants' claim 1. As the Examiner is surely aware, to establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Thus, the Examiner has failed to provide a *prima facie* rejection of claim 1.

In the Response to Arguments, the Examiner states, "[i]n response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references." However, contrary to the Examiner's assertion, Applicants' arguments were in fact against the *combination* of art cited by the Examiner. For example, Applicants' previous response (filed January 23, 2006) stated, "Montero in view of Goldick does not teach or suggest a distributed store configured to provide locked access to the primary

state of session data to a process executing within one or a plurality of application servers, ...". Additionally, Applicants' previously argued (as above), "the Examiner's proposed combination of Montero and Goldick would not result in a system that included a distributed store configured to send a lock token to a process executing within one or a plurality of application servers." Any statements or arguments regarding individual references where intended to show that the Examiner's reliance on a particular reference was misplaced.

Also, the Examiner has failed to provide a proper motivation for combining Montero and Goldick. The Examiner states that it would have been obvious to combine the teachings of Montero with those of Goldick "in order to prevent data inconsistency." However, preventing data inconsistency would only provide motivation for one to use either of systems taught by Montero or Goldick individually, as they both already individually provide methods to prevent data inconsistency. Thus, the Examiner's reason does not provide any motivation to combine or modify the individual teachings of Montero and/or Goldick. Additionally, as is well known in the art, there are many different ways to prevent data inconsistency. The mere desire to prevent data inconsistency would not in any way suggest or motivate one to modify the teachings of Montero with those of Goldick. As held by the U.S. Court of Appeals for the Federal Circuit in *Ecolochem Inc. v. Southern California Edison Co.*, an obviousness claim that lacks evidence of a suggestion or motivation for one of skill in the art to combine prior art references to produce the claimed invention is defective as hindsight analysis. In addition, the showing of a suggestion, teaching, or motivation to combine prior teachings "must be clear and particular." "Broad conclusory statements regarding the teaching of multiple references, standing alone, are not 'evidence'." *In re Dembiczak*, 175 F.3d 994, 50 USPQ2d 1614 (Fed. Cir. 1999).

In the Response to Arguments, the Examiner points out that motivation to combine reference may come from the references themselves, from knowledge of those skilled in the art or from the nature of the problem to be solved. The Examiner also refers to various features of the cited art that are relied on in the rejection of claim 1. The

Examiner then repeats the assertion, "based on Montero in view of Goldick, it would have been obvious ... to utilize the teaching of Goldick to the system of Montero in order to prevent data inconsistency." The Examiner has not provided any rebuttal or additional argument regarding Applicants' argument that the desire to prevent data inconsistency would not in any way suggest or motivate one to modify the teachings of Montero with those of Goldick. The Examiner's reasoning appears to be that since Montero teaches a common session database in a distributed environment and since Goldick teaches a resource lock management scheme in order to prevent the "lost update" problem, it would have been obvious to combine the Montero and Goldick. **However, just pointing out the individual features of two respective references does not provide any motivation to combine the references.**

Furthermore, the Examiner is improperly focusing his arguments only on specific differences between the present invention and the prior art, instead of considering the invention as a whole. The Examiner is arguing the obviousness of including the system for providing client computers with locked access to shared resources as taught by Goldick with the periodic updating by application servers of session data as taught by Montero. However, "the question under 35 U.S.C. 103 is not whether in differences themselves would have been obvious, but whether the claimed invention as a whole would have been obvious." *Stratoflex, Inc. v. Aeroquip Corp.*, 713 F.2d 1530, 218 USPQ 871 (Fed. Cir. 1983); *Schenck v. Nortron Corp.*, 713 F.2d 782, 218 USPQ 698 (Fed. Cir. 1983). *See also*, M.P.E.P 2141.02. "Most if not all inventions arise from a combination of old elements. *See In re Rouffet*, 149 F.3d 1350, 1357, 47 USPQ2d 1453, 1457 (Fed. Cir. 1998). Thus, every element of a claimed invention may often be found in the prior art. *See id.* However, identification in the prior art of each individual part claimed is insufficient to defeat patentability of the whole claimed invention. *See id.*" *In re Werner Kotzab*, 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000) (*emphasis added*).

Thus, for at least the reasons above, the rejection of claim 1 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 17, contrary to the Examiner's assertion, Montero in view of Goldick does not teach or suggest means for providing locked access to the primary state to a process executing within one of a plurality of application servers, wherein the means for providing locked access comprises means for sending the process a lock token, wherein only processes that have received a lock token can access the primary state. The Examiner does not provide a separate rejection for claim 17, but instead relies upon the rejection of claim 1, discussed above.

As noted above regarding the rejection of claim 1, the Examiner admits that Montero does not teach a distributed store configured to send a lock token and relies upon Goldick, citing paragraphs 24-25 and 44-45. Goldick teaches a system for managing allocation of resources and locks to client computer systems. Goldick's system includes locking resources and maintaining subscriptions to lock-related events to effectively allow for asynchronous grants of a lock based on the time of the request to alleviate lock starvation. Goldick further teaches that a server may break an existing lock to prevent lost resources.

However, Goldick, even if combined with Montero, does not teach anything about means for sending lock tokens to processes *executing within an application server.* Instead, as noted above regarding claim 1, Goldick teaches a server that provides requesting *client* computer systems with locked access to resources. Goldick specifically refers to managing access by client computer systems. Goldick's system for managing access to shared resources by client computer systems is very different from a sending a lock token to a process executing within an application server and from Montero's system for providing periodic storing of session data by multiple application servers. Goldick does not mention anything about sending a lock token to processes executing within an application server. Nor does Goldick have anything to do with providing a plurality of application servers locked access to session data.

Additionally, as described above regarding the rejection of claim 1, Goldick is not directed toward providing locked access to *session data* at all. Instead, Goldick teaches a

system for managing *client* access to shared resources, such as "text documents, application program modules, data objects, properties or attributes for data objects". The shared resources taught by Goldick are very different from session data configured for access by a plurality of application servers. The Examiner has not cited any prior art, either singly or combination, that teaches or suggests a means for providing locked access to the primary state of session data to a process executing within one of a plurality of application servers, where the means for providing locked access comprising means for sending the process a lock token, wherein only processes that have received a lock token can access the primary state.

Additionally, the Examiner's proposed combination of Montero and Goldick would not result in a system that included a means for sending a lock token to a process executing within one of a plurality of application servers. Instead, the Examiner's combination of Montero and Goldick would result in Montero's back-end database system that includes application servers maintaining session data for clients as taught by Montero, but also allows clients locked access to a server database, based on providing requesting clients lock tokens, as taught by Goldick. Please refer to the remarks above regarding the rejection of claim 1 for a more detailed discussion of why the Examiner's proposed combination of Montero and Goldick would not result in a system that included means for sending a lock token to a process executing within one of a plurality of application servers.

Also, as noted above regarding the rejection of claim 1, the Examiner has failed to provide a proper motivation for combining Montero and Goldick. Please refer to Applicants' remarks above regarding the rejection of claim 1 for a detailed discussion of the Examiner's failure to provide proper motivation to combine Montero and Goldick.

Thus, for at least the reasons above, the rejection of claim 17 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 20, contrary to the Examiner's assertion, Montero in view of Goldick does not teach or suggest granting a lock to a process requested locked access to the primary state of session data, wherein said granting comprises sending a lock token to the process, wherein only processes that have received a lock token can access the primary state. The Examiner does not provide a separate rejection for claim 20, but instead relies upon the rejection of claim 1, discussed above.

As noted above regarding the rejection of claims 1 and 17, the Examiner admits that Montero does not teach a distributed store configured to send a lock token and relies upon Goldick, citing paragraphs 24-25 and 44-45. As noted above, Goldick teaches a system for managing allocation of resources and locks to client computer systems. Goldick's system includes locking resources and maintaining subscriptions to lock-related events to effectively allow for asynchronous grants of a lock based on the time of the request to alleviate lock starvation. Goldick further teaches that a server may break an existing lock to prevent lost resources.

However, as described above regarding claim 1 and 17, Goldick, even if combined with Montero, does not mention anything about sending lock tokens to processes *executing within an application server*. Instead, as noted above regarding claims 1 and 17, Goldick teaches a server that provides requesting *client* computer systems with locked access to resources. Goldick specifically refers to managing access by client computer systems. Goldick's system for managing access to shared resources by client computer systems is very different from a sending a lock token to a process executing within an application server and from Montero's system for providing periodic storing of session data by multiple application servers. Goldick does not mention anything about sending a lock token to processes executing within an application server. Nor does Goldick have anything to do with providing a plurality of application servers locked access to session data.

Additionally, as described above regarding the rejections of claims 1 and 17, Goldick is not directed toward providing locked access to *session data* at all. Instead,

Goldick teaches a system for managing *client* access to shared resources, such as "text documents, application program modules, data objects, properties or attributes for data objects". The shared resources taught by Goldick are very different from session data configured for access by a plurality of application servers. The Examiner has not cited any prior art, either singly or combination, that teaches or suggests granting a lock to a process requesting locked access to the primary state of session data, wherein said granting comprises sending a lock token to the process, wherein only processes that have received a lock token can access the primary state.

Additionally, the Examiner's proposed combination of Montero and Goldick would not result in a system that included sending a lock token to a process executing within one of a plurality of application servers. Instead, the Examiner's combination of Montero and Goldick would result in Montero's back-end database system that includes application servers maintaining session data for clients as taught by Montero, but also allows clients locked access to a server database, based on providing requesting clients lock tokens, as taught by Goldick. Please refer to the remarks above regarding the rejection of claim 1 for a more detailed discussion of why the Examiner's proposed combination of Montero and Goldick would not result in a system that included means for sending a lock token to a process executing within one of a plurality of application servers.

Also, as noted above regarding the rejection of claim 1, the Examiner has failed to provide a proper motivation for combining Montero and Goldick. Please refer to Applicants' remarks above regarding the rejection of claim 1 for a detailed discussion of the Examiner's failure to provide proper motivation to combine Montero and Goldick.

Thus, for at least the reasons above, the rejection of claim 20 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks also apply to claim 31.

The Examiner rejected claims 26, 28-30, 35 and 37 as being unpatentable over Montero in view of Bennett (U.S. Patent 5,734,909) and Bender, et al. (U.S. Publication 2003/0163494) (hereinafter "Bender"), and further in view of Eshel, et al. (U.S. Publication 2003/0018785) (hereinafter "Eshel").

Regarding claim 26, Montero in view of Bennet, Bender and Eshel fails to teach or suggest requesting the process release the locked access by the distributed store and the process releasing the locked access in response to the request by the distributed store. Montero does not teach any sort of locking mechanism, as admitted by the Examiner. Bennet teaches a system in which client requests requiring locked access to a resource already locked are queued until the locked resource is available (Bennet, column 3, line 62 – column 4, line 14). Bender teaches a system in which processes and threads requiring locked access to already locked resources must wait until the process holding locked access to the resource has finished with the resource. The Examiner admits that Montero, Bennet and Bender, either singly or in combination, fail to teach or suggest a distributed store configured to request the process to release the locked access, wherein the process is configured to release the locked access in response to the request.

The Examiner relies upon Eshel, citing paragraphs [0011] and [0012], as well as Fig. 2. However, Eshel does not teach or suggest a distributed store requesting a process release a locked access. Instead, Eshel teaches that the other node requiring locked access, not the server or distributed store, sends a request to a process that holds a lock token for a shared resource. Specifically, at the Examiner's cited passage, Eshel teaches that when the token server receives a request from a node for a token that is current held by another node, the other node needs to relinquish its token and that "[t]his is accomplished by having the lock manager *of the requesting node* send a revoke request to the node holding the token" (Eshel, paragraph [0012]). Thus, Eshel plainly teaches that the requesting node, not the token server or a distributed store, sends a request a process to release a locked access. Additionally, by teaching that the requesting node sends a revoke request to a node that holds a lock token for a shared resource, **Eshel teaches away** from a distributed store configured to request a process to release a locked access.

Also Eshel fails to teach or suggest that the process holding a locked access is configured to release the locked access in response to a request *from a distributed store*. Instead, Eshel teaches that the node holding a lock token, "checks whether a lock requiring the token is currently held, *waits for any such lock to be released*, and then sends a message to the token server to relinquish the token" (Eshel, paragraph [0012]). Thus, Eshel clearly teaches that a process holding locked access does not release the locked access in response to a request to release the lock access. Eshel's nodes wait until the lock is released.

Thus, Eshel fails to teach the functionality for which the Examiner relies on Eshel. Since Montero, Bennet, Bender and Eshel all fail to teach or suggest the distributed store requesting the process to release the locked access, wherein the process is configured to release the locked access in response to the request from the distributed store, the Examiner's combination of Montero, Bennet, Bender and Eshel also fails to teach or suggest such functionality.

Furthermore, the Examiner has failed to provide a proper motivation to combine the teachings of Eshel with those of Montero, Bennet and Bender. The Examiner states that it would have been obvious to combine the teachings of Eshel with those of Montero "in order to provide the locked access to another node." However, the Examiner stated motivation is merely a reason why one would use the system taught by the background section of Eshel. The desire to provide locked access to another node would not motivate one to modify the teachings of Montero, Bennet or Bender. Additionally, the Examiner's stated motivation amounts to nothing more than stating what the Examiner hopes to achieve by making the combination. Such a conclusory statement does not provide motivation to modify the existing systems of Montero, Bennet and Bender.

In the Response to Arguments, the Examiner fails to rebut or address Applicants' argument regarding a lack of proper motivation to combine the teachings of Eshel with those of Montero, Bennet and Bender.

For at least the reasons below, the rejection of claim 26 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks apply to claim 35.

The Examiner also rejected claims 2, 3, 22 and 32 under 35 U.S.C. § 103(a) as being unpatentable over Montero in view of Goldick, and further in view of Eshel, claims 4, 7, 8, 19, 23-25, 33 and 34 as being unpatentable over Montero in view of Goldick and further in view of Bennett, claims 5, 18 and 21 as being unpatentable over Montero in view of Goldick, and further in view of Bender, and claims 26, 28-30, 35 and 37 as being unpatentable over Montero in view of Bennett and Bender, and further in view of Eshel. Applicants respectfully traverse this rejection of these claims for at least the reasons presented above regarding their respective independent claims.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

Claims Objected To But Otherwise Allowable:

Claim 10 was objected to, but would be allowable if rewritten or amended to overcome the rejections under 35 U.S.C. § 112, second paragraph. Applicants submit that claims 10, 11 and 13-16 are in condition for allowance.

CONCLUSION

Applicants submit the application is in condition for allowance, and prompt notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-11700/RCK.

Also enclosed herewith are the following items:

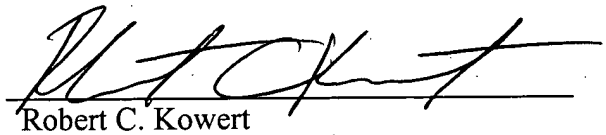
☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: June 16, 2006